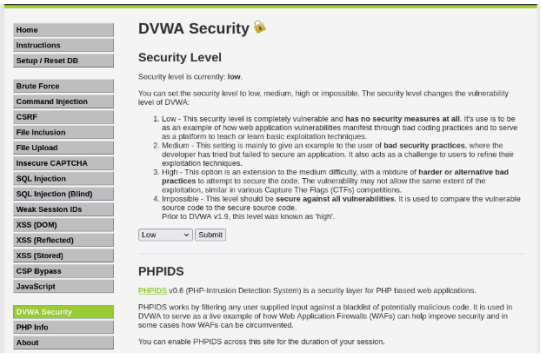
Explotación Controlada de una Inyección SQL y Validación Manual de una Vulnerabilidad con Burp Suite.

1. Objetivo del ejercicio

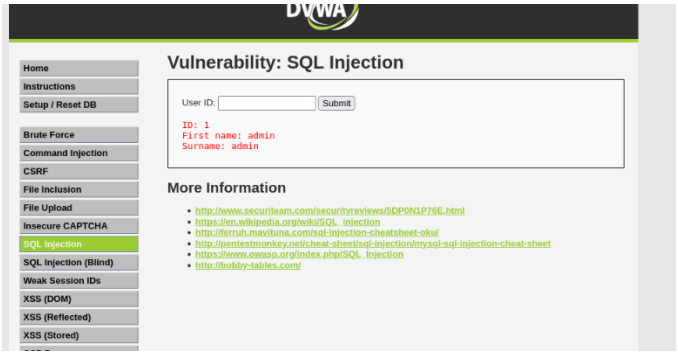
El objetivo fue simular una prueba de penetración intermedia sobre la aplicación web vulnerable DVWA. Se utilizó Burp Suite para identificar, validar y documentar una vulnerabilidad de tipo SQL Injection, determinando el riesgo potencial de manera profesional y controlada.

2. Configuración y Reconocimiento

- **Configuración de DVWA**: Se inició la aplicación web DVWA en un entorno local con Docker. El nivel de seguridad fue configurado en **Low**.

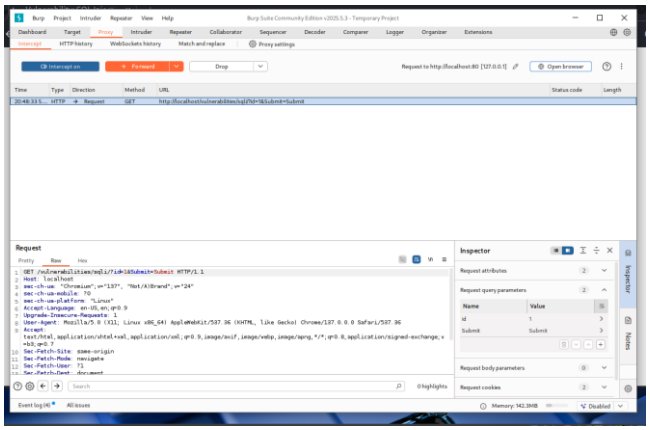


- **Configuración de Burp Suite**: Se configuró el navegador para redirigir todo el tráfico por el proxy de Burp Suite, en la dirección 127.0.0.1:8080.
- **Módulo de SQL Injection**: Se navegó al módulo de SQL Injection para iniciar las pruebas.



3. Interceptar y analizar la solicitud vulnerable

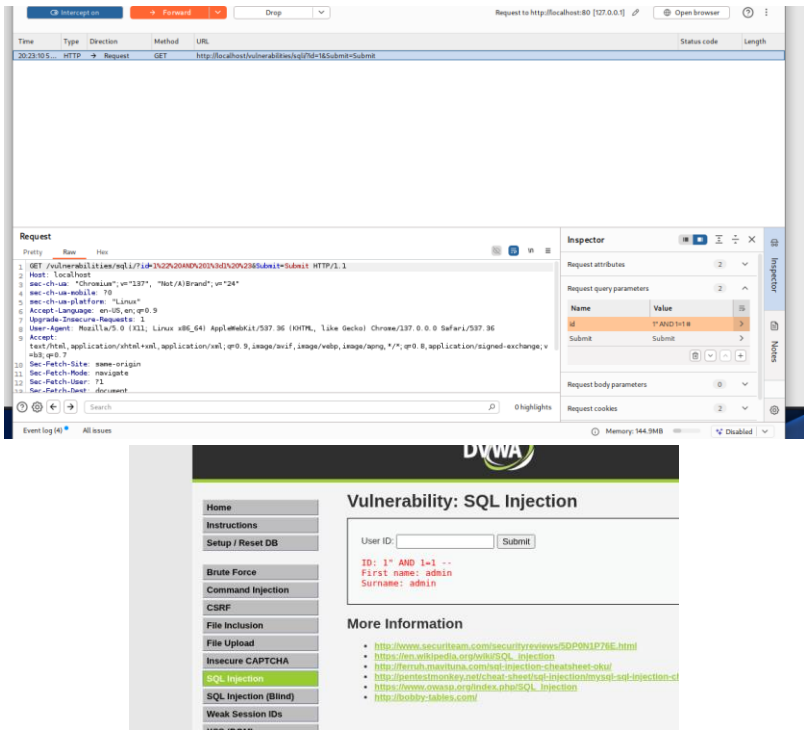
Se utilizó Burp Suite para interceptar la solicitud HTTP generada al ingresar 1 en el campo "User ID" de DVWA. El parámetro identificado como susceptible a la inyección fue el parámetro **id**.



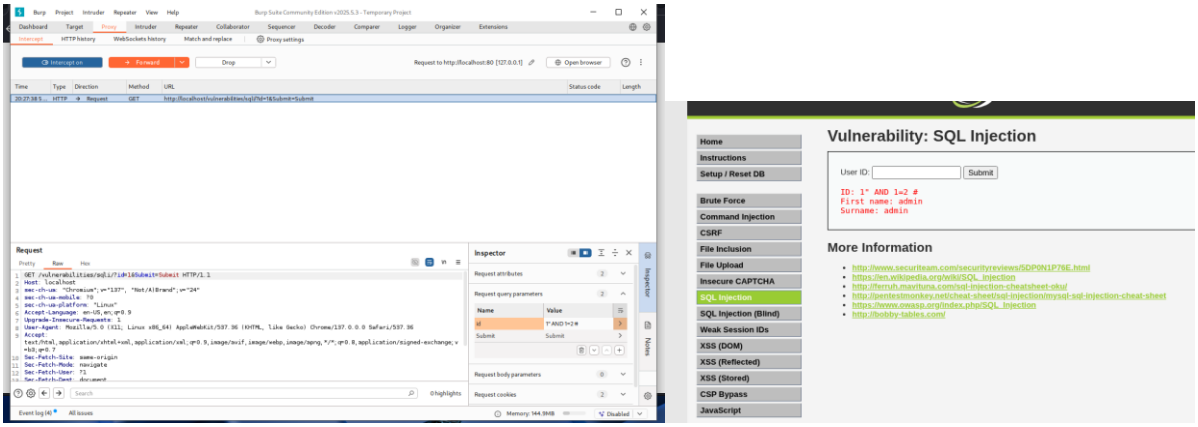
4. Probar la inyección SQL manual

Se probaron varios payloads para confirmar la vulnerabilidad. La primera prueba con 1' AND 1=1 -- resultó en un error de sintaxis SQL, confirmando la falta de validación de la entrada. Tras ajustar la sintaxis, se realizaron las siguientes pruebas:

- **Payload verdadero:** Se utilizó 1" AND 1=1 #. La aplicación devolvió un resultado, demostrando que la condición era verdadera.



- **Payload falso:** Se utilizó 1" AND 1=2 #. La aplicación no devolvió resultados, demostrando que la condición era falsa.

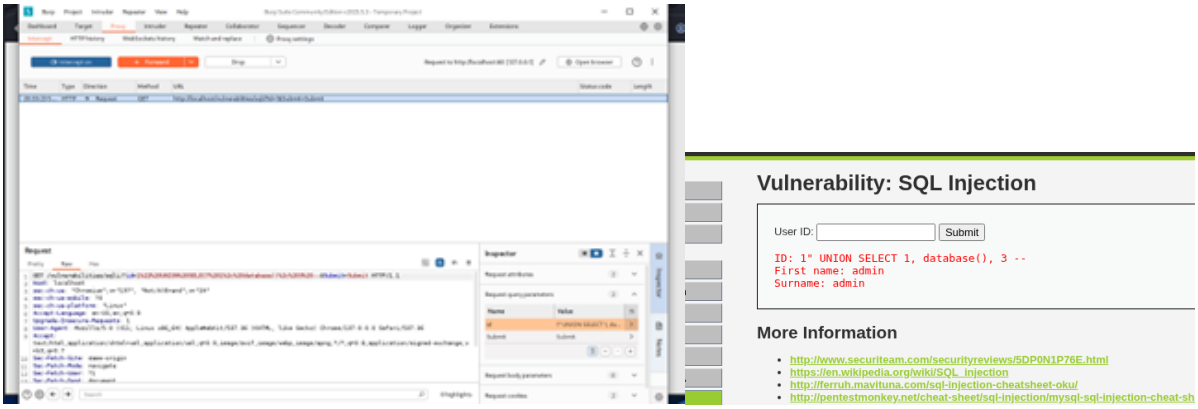


La diferencia en los resultados entre una condición verdadera y una falsa confirmó la existencia de la vulnerabilidad de inyección SQL.

5. Validar el impacto de la vulnerabilidad

Para demostrar el impacto, se utilizó un payload UNION SELECT para extraer el nombre de la base de datos de la aplicación.

- **Payload utilizado:** 1" UNION SELECT 1, database(), 3 --.
- **Resultado:** El servidor procesó la consulta UNION SELECT y devolvió el resultado admin. El hecho de que la consulta se haya ejecutado con éxito demuestra que es posible extraer información de la base de datos. En este caso, el nombre de la base de datos es **dvwa**.



6. Documentar hallazgos

Hallazgos:

- **Parámetro vulnerable:** id en la URL de la página de SQL Injection.
- **Payloads utilizados:** 1" AND 1=1 # y 1" UNION SELECT 1, database(), 3 --.
- **Resultados observados:** Se demostró que la aplicación es susceptible a una inyección SQL por medio de la manipulación del parámetro id. Se logró ejecutar una consulta para extraer el nombre de la base de datos.
- **Riesgo estimado: Alto**. La vulnerabilidad permite a un atacante no autorizado extraer información sensible de la base de datos, lo que podría conducir a la divulgación de datos de usuarios, credenciales y otra información crítica.

7. Recomendación profesional:

Para mitigar esta vulnerabilidad, se recomienda encarecidamente utilizar consultas preparadas (Prepared Statements) con parámetros. Esta técnica separa la lógica SQL de los datos del usuario, lo que previene que los caracteres maliciosos (", ') sean interpretados como parte de la consulta. Además, se sugiere utilizar un "Web Application Firewall" (WAF) como medida de defensa en profundidad para filtrar las entradas sospechosas.

8. Reflexión Final:

- **Comportamiento de la aplicación:** El comportamiento de la aplicación cambia drásticamente con cada payload. Un payload de sintaxis correcta, aunque malicioso, es procesado sin errores, mientras que un payload con un error de sintaxis causa que el servidor de la base de datos falle y arroje un mensaje de error público.
- **Facilidad de explotación:** Sería relativamente fácil para un atacante obtener datos confidenciales. La falta de validación de entrada permite la ejecución de comandos SQL arbitrarios, lo que podría comprometer toda la base de datos.
- **Importancia de la validación:** Es fundamental validar cada parámetro de entrada del usuario. Este ejercicio demuestra que incluso un simple campo de búsqueda puede ser una puerta de entrada para un ataque. La validación y la sanitización son cruciales para mantener la integridad y confidencialidad de la información.